Ex. M

HIGHLY CONFIDENTIAL – ATTORNEY'S EYES ONLY

UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

ELASTICSEARCH, INC. AND ELASTICSEARCH B.V.,

Plaintiffs,

- against -

FLORAGUNN GmbH,

Defendant.

Case Nos. 4:19-cv-05553-YGR and 4:20-cv-07514-YGR

EXPERT REPLY REPORT OF DR. MARTIN WALKER

July 23, 2021

certain floragunn source code is substantially similar to and copied from certain Elastic source code, certain floragunn source code is a derivative work of certain Elastic source code, and that floragunn accessed Elastic source code through a variety of methods including reverse compilation of certain Elastic binaries. With respect to efficiency constraints or concerns, Dr. Astrachan fails to show that those concerns or constraints so narrow the practical range of expression that doctrines limiting the scope of copyright protection must be applied. With respect to common programming practices and techniques, Dr. Astrachan fails to establish that use those practices or techniques are practically the only way to express the relevant code in that context rather than an expressive choice. With respect to Dr. Astrachan's arguments regarding functional versus expressive similarities, Dr. Astrachan generally fails to account for the expressive similarities I note in my opening report (and, I would note, functional similarities between infringed and infringing code are to be expected).

5. Additionally, Dr. Astrachan does not dispute my identification of accused floragunn code or derivative works of the accused floragunn code in AESS and ODFE. I note that I maintain my opinion on that subject.

III. Reply to Dr. Astrachan's Report

A. Decompilation and Access

6. As Dr. Astrachan acknowledges, Uri Boness testified that Hendrik Saly told him that he had decompiled Elastic's binaries. In my opening report I relied on Mr. Boness's testimony to support my opinions that floragunn had accessed certain Elastic source code via such reverse compilation.

7. Reverse compilation is a feasible means for accessing the underlying source code.

8. It is not controversial that floragunn could and did reverse compile the code:

a. "Could"—even Dr. Astrachan shows it is possible. As explained by Dr. Astrachan in ¶ 181, he performed such reverse compilation.

    i. Thus not only is it technically possible, it would be straight-forward for floragunn's developers to perform such reverse compilation. I have personal experience performing such reverse compilation and I know it is straight-forward.

b. "Did" —supported by Uri Boness's testimony, the similarities in the source code, the undisputed availability of the Elastic binaries, and the indicia of reverse compilation I note in my opening report.

c. Note that Hendrik Saly is the author of the accused floragunn Java source code (*see* Jochen Kressin March 9, 2021 Dep. Ex. 185), and Saly has admitted that he reverse compiled Elastic binaries.

9. ¶ 50: That reverse compilation can provide access is not controversial. I note Henrik Saly, who is the author of the referenced in ¶ 50 and 51 (*see* Jochen Kressin March 9, 2021 Dep. Ex. 185), had experience reverse compiling Elastic code.

10. ¶ 51: Access + substantial similarity are sufficient to show copying. Access is not speculative. My opinion shows substantial similarity.

11. Also in my opening report, I noted certain indicia of a reverse compilation process evident in the floragunn source code (*see* my opening report ¶ 227 and ¶ 233-240.)

12. Elastic Shield and X-pack binaries are and were widely distributed via the internet. In fact, Dr. Astrachan was able to personally download such binaries (see Astrachan ¶ 227 and f/n 88).

13. Given the evidence specified above, it is clear that Dr. Astrachan's assertion that I didn't identify any non-speculative evidence of decomplication is simply incorrect.[1]

B. getLiveDocs()

14. I have been informed that, for efficiency considerations to require filtration of a source code element, efficiency considerations must so narrow the practical range of choice that protecting the expression would effectively accord protection to the idea itself. Blindly citing to efficiency considerations does not show that the range of possible expression is limited.

15. The use of "lazy" evaluation (such as through the anonymous class in the Elasticsearch source code) does not always result in improved efficiency. Indeed Dr. Astrachan admits as much in ¶ 54. As Dr. Astrachan explains "This particular performance tradeoff makes particular sense if there may be very large document segments, especially if it is likely that there may never be a need to know the status of many documents in the segment."

16. Accordingly, there are different methods to implement this function. For example, consider the previous floragunn implementation (i.e. Commit ID 8be1c2c3a12114c1659e9111a13fc792a276a4e2): shown below. As can be seen, the relevant quantity "liveDocs" is calculated up front in the constructor, as opposed to the "lazy" calculation in the Elastic source code that was copied by floragunn.

```
 80     DlsFlsFilterLeafReader(final LeafReader delegate, final Set<String>
includesExcludes, final Query dlsQuery) {
…
179                 if (in.hasDeletions()) {
180                     final Bits oldLiveDocs = in.getLiveDocs();
181                     assert oldLiveDocs != null;
182                     final DocIdSetIterator it = new BitSetIterator(bits,
0L);
```

[1] I note that Dr. Astrachan does not dispute my opinion that floragunn accessed Elastic's publicly available source code, which is supported by ample evidence. (*See* March 1, 2021 Deposition of Jochen Kressin 14:6-16:19, 155:14-160:10; *id.* Exs. 160, 171; March 9, 2021 30(b)(6) Deposition of Jochen Kressin 152:14-155:16, 155:17-157:4, 157:5-171:7, 171:8-173:21, 188:20-191:13, 173:22-184:24, 184:25-188:8, 195:20-199:23, 199:24-206:15; *id.* Exs. 192, 193, 194, 195, 196, 198, 199; April 22, 2021 Rule 30(b)(6) Deposition of Jochen Kressin 529:15-530:18, 530:19-532:16, 532:17-535:8, 535:9-539:14; *id.* Exs. 169, 311, 312, 313.)

did not identify any aspects of the Elastic code that should have been filtered out of the comparison step.

93. Accordingly, comparison of the properly filtered Elastic source code file with the accused floragunn source code yields substantial amounts of copied source code, even granting Dr. Astrachan's arguments in ¶ 165-171 (which I dispute). At best, Dr. Astrachan's identification of alternative antecedents to floragunn's accused code provides a partial, fragmentary and convoluted explanation of floragunn's accused code. For example KUI does not provide a satisfactory explanation for the lines of index.html that I cite at ¶ 219-221 in my opening report. Accordingly, nothing in Dr. Astrachan's argument changes my opinion that the referenced floragunn code is substantially similar to the copyrighted Elastic source code or derived from the Elastic source code.

94. ¶ 169: Dr. Astrachan has two specific observations regarding the copied source code outside of the scope the KUI code. First, he incorrectly asserts that the pipe syntax is not protectable since it is a standard syntax. However, my analysis focused on the specific expression at line 13, not the pipe syntax in general. Second, he alleges that "combining filters with a length check is not unusual." Regardless of whether the expression is unusual, it does represent expression substantially similar to the Elastic source code expression. Finally, Dr. Astrachan opines that unspecified additional code is "too minor to be protected expression." The code I identified in my filtration analysis is protectable expression. Should Dr. Astrachan supplement his opinion to add specifics to this analysis, I reserve my right to comment.

L. bulk_op_type

95. ¶ 175: As I explained in my opening report, the choice to compare the variable action to the constant TransportShardBulkAction.ACTION_NAME as well as the variable names represent protectable content. Dr. Astrachan seems to ignore these aspects.

96. ¶ 176: I disagree: since it is my opinion that the statement was created using a reverse compilation process rather than by an expressive choice by a floragunn developer, the resulting floragunn code is based on the Elastic source code.

97. ¶ 177: Dr. Astrachan seems to argue that the similarities are a result of independent development, thus ignoring the evidence I cited that shows evidence of reverse compilation. Further, even the code cited by Dr. Astrachan could have been created through a reverse compilation process. I note that Dr. Astrachan's experiments with decompilation confirm my opinion that decompilation is quite straightforward.

98. ¶ 178: I disagree. I explained in detail the bases for my conclusions regarding reverse compilation of the code.

99. ¶ 179-180: Dr. Astrachan seems to agree that, at least in some cases, the compilation-reverse compilation process may reorder the tests. ("decompilation of the binary could produce code with a different order than the order in the original code").

100. ¶ 181: Dr. Astrachan assumes, without evidence, that a particular decompiler was used by floragunn. As Dr. Astrachan admits, other decompilers may give different results. Further, I note that Dr. Astrachan does not identify with specificity the decompiler he used to create the decompiled code.[7] Nor does Dr. Astrachan state that the decompiler he used was available at the time floragunn decompiled the code or would have been the logical choice for Hendrik Saly to use. Should Dr. Astrachan remedy these ambiguities, I specifically reserve my right to further comment on his analyses.

[7] Dr. Astrachan references "the standard decompiler that is part of the IntelliJ IDE." However, I was unable to identify the "standard" decompiler; there are at least two decompilers associated with IntelliJ JDE. Neither are identified as the "standard," and neither appear to reproduce the precise results claimed by Dr. Astrachan.

101. ¶ 182: The alleged discrepancies between the literal output of Dr. Astrachan's decompilation process and the floragunn source code may be explained by either differences between Dr. Astrachan's process and the actual process used by floragunn to decompile the code, or simple edits made by floragunn in integrating the decompiled results into the java file, or both. In any case these alleged discrepancies do not affect my opinion regarding the likely origin of the source code.

102. ¶ 184-186: Dr. Astrachan seems to ignore a likely explanation for the discrepancies: floragunn developers made simple edits in the process of integrating the decompiled code into the java file. Additionally, I note that it is unlikely that a developer would use the variable ar in the manner suggested by Dr. Astrachan as the variable ar would then be used for two purposes: to refer to a specific action request (lines 1243 and 1251) as well as action requests generally (at line 392).

103. ¶ 187-188: Dr. Astrachan's speculations relating to the sequential process of the modifications are without basis. I note that Dr. Astrachan points to no support from floragunn developers for such a process, nor does he point to any other instance of incremental commits. To the contrary I found many instances of commits in the floragunn git repos that do not follow his speculative process. I have listed several of these below:

- repo: searchguard commit ID: cf3d3c03cdfa9a554a416a72a655d47f58645406;
- repo: searchguard commit ID: 2549fbd6baa1165c4486ff790f8c2c918d602d23;
- repo: searchguard commit ID: b00179fcd51f5d53dba705abf1eb1ef7c6abfaad;
- repo: searchguard-module-dlsfls commit ID: 78b6c9b445b0d79c910515aa2d6530e156879065;

- repo: searchguard-enterprise-modules commit ID: cea230df39ea9b17cbb25ef244a5641a3da81266;
- repo: searchguard-ssl commit ID: b0699f5943480d6debeb96bd518d838ae582c5d9;
- repo: searchguard-module-dlsfls commit ID: 1c81074c3a26291c9c5d5806d5fbc6b47a0bad8a

M. Update with DLS

104. As an initial matter, I note that Dr. Astrachan does not disagree with my filtration analysis.

105. ¶ 192: Dr. Astrachan opines that the "high level differences are enough to support my opinion that an ordinary, reasonable computer programmer would find these methods very different in total concept and feel." I disagree. At a high level they perform the same function in the same way to prevent bulk update requests in the context of document level security.

106. ¶ 192, 195, 197, 198: I disagree with Dr. Astrachan's characterization of the identified source code. The accused code is related to processing of bulk requests in the context of document level security. Dr. Astrachan identifies source code outside the scope of my analysis and notes that this code is not copied. In doing so, he seems to ignore the relevant floragunn source code that is substantially similar to the Elastic source code.

107. ¶ 194: As I explained in my opening report, the expressive content of the loop statements are substantially similar. There are many methods in the java programming language to iterate over items in a list, including while statement and ArrayList operators. Thus the use of a for statement to iterate over the same request items represents substantial similarity.

Dr. Astrachan do not change my opinion that the floragunn code is substantially similar to the identified Elastic code.

P. Works as a Whole

119. ¶ 219: I disagree with Dr. Astrachan that the works as a whole are not substantially similar. As I noted in my opening report, the multiple instances of floragunn code that are substantially similar to or derived from Elastic code are qualitatively important to the works as a whole. I disagree that the similarities I have identified are "isolated examples of similarity."

Q. Amazon Web Services

120. ¶ 220: I note that Dr. Astrachan does not dispute my analysis of the presence of copies of the accused floragunn source code or their derivative works in the AESS and ODFE source code. As stated in my opening report, my analysis of the instances of accused floragunn source code or their derivative works found in AESS and ODFE are the same as my opinions regarding the corresponding accused floragunn source code.

IV. Signature

By: 

Dr. Martin Walker

Date: July 23, 2021